Idea 1: Block Dodger

Main Idea:

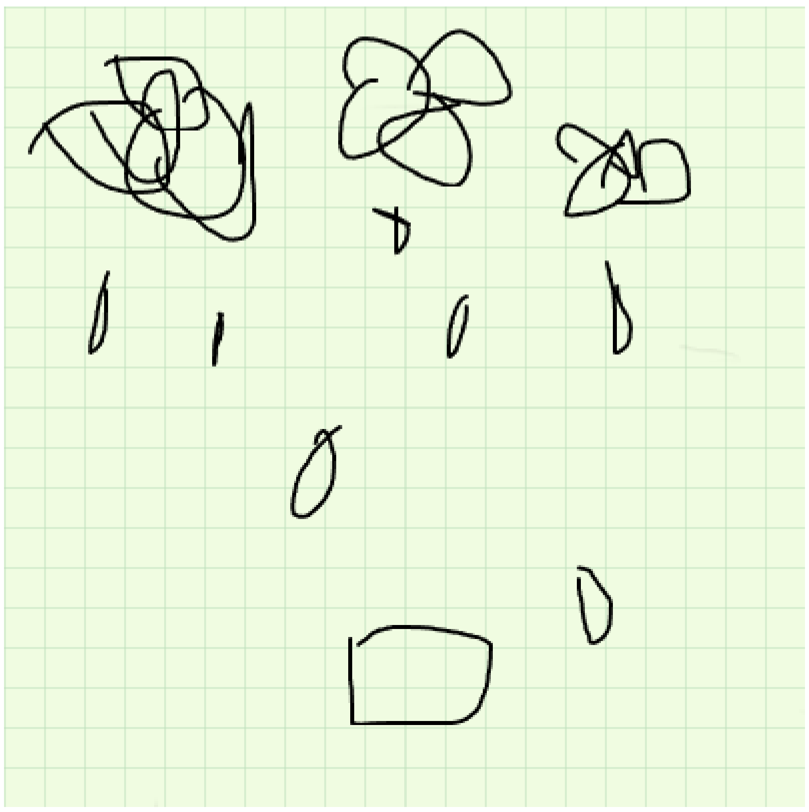
A game where you control a character that must dodge falling blocks. The goal is to avoid them as long as possible, with the difficulty increasing over time.

Interactions:

- **Movement:** Use the arrow keys (or AD) to move the character left and right.
- **Objective:** Avoid falling blocks and survive as long as possible.
- **Collision:** Colliding with the various blocks

Main Design Elements:

- **Character:** A simple square that moves left and right on the screen.
- **Falling Blocks:** Randomly generated blocks that fall from the top of the screen. Their speed and amount increase as the game progresses.
- **Background:** A simple gradient or solid color with no distractions.
- **Game Over:** The game ends when the character collides with a block.



Idea 2: Color Match Runner

Main Idea:

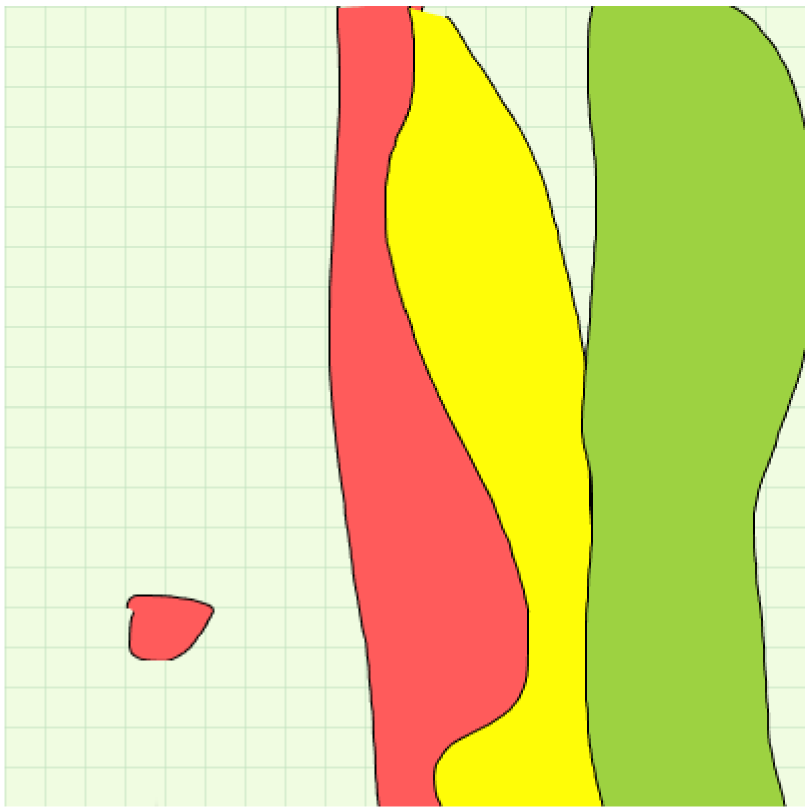
A simple side-scrolling runner game where you control a character that must switch colors to match the obstacles coming at them.

Interactions:

- **Movement:** Use the arrow keys to jump and duck.
- **Switch Colors:** Press spacebar to switch the character's color to match incoming obstacles.
- **Objective:** Avoid obstacles that don't match your character's current color.

Main Design Elements:

- **Character:** A small circle that changes color when you press the spacebar.
- **Obstacles:** Obstacles of different colors that the player must match.
- **Background:** Simple scrolling background with increasing speed.
- **Game Over:** The game ends when the character hits an obstacle that doesn't match.



Idea 3: Simple Maze Runner

Main Idea:

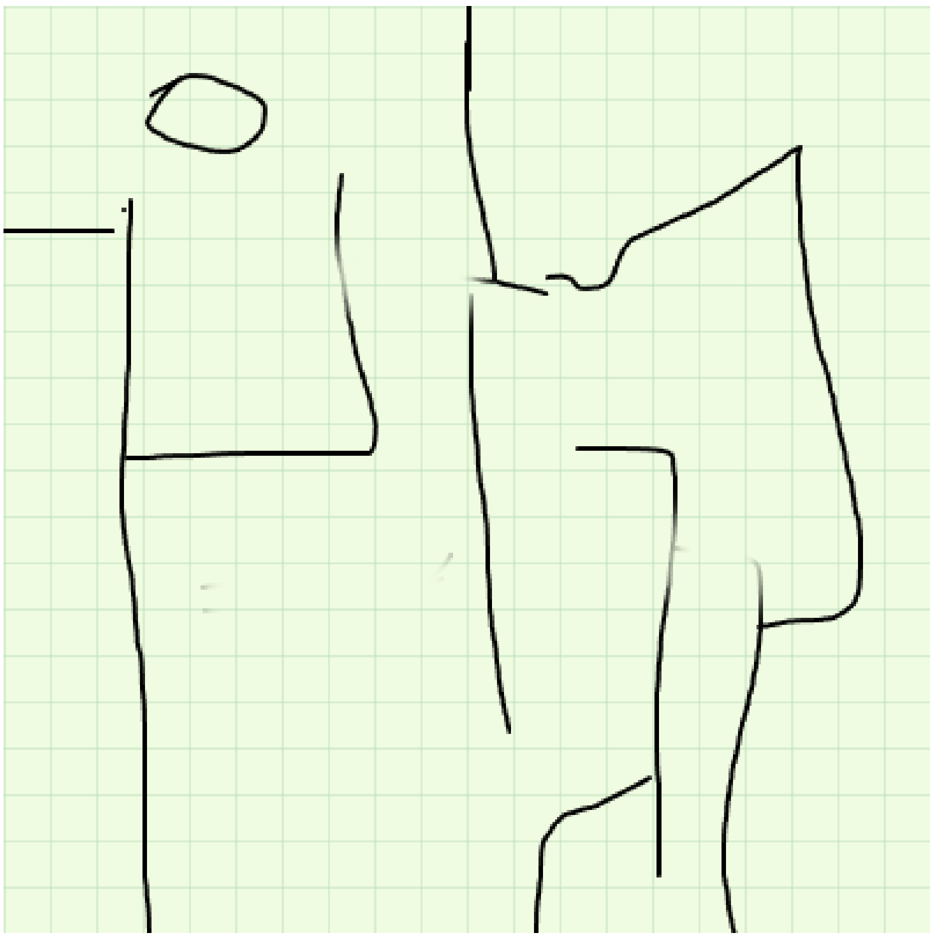
A basic maze navigation game where the player needs to find the exit of a randomly generated maze.

Interactions:

- **Movement:** Use the arrow keys (or WASD) to move the player through the maze.
- **Objective:** Reach the exit of the maze before time runs out.

Main Design Elements:

- **Maze Generation:** Randomly generate a simple maze using an algorithm that places walls and an exit.
- **Player:** A small square or circle representing the player.
- **Exit:** A defined point in the maze that the player must reach.
- **Timer:** The player has a limited time to reach the exit.



Idea 4: Falling Words

Main Idea:

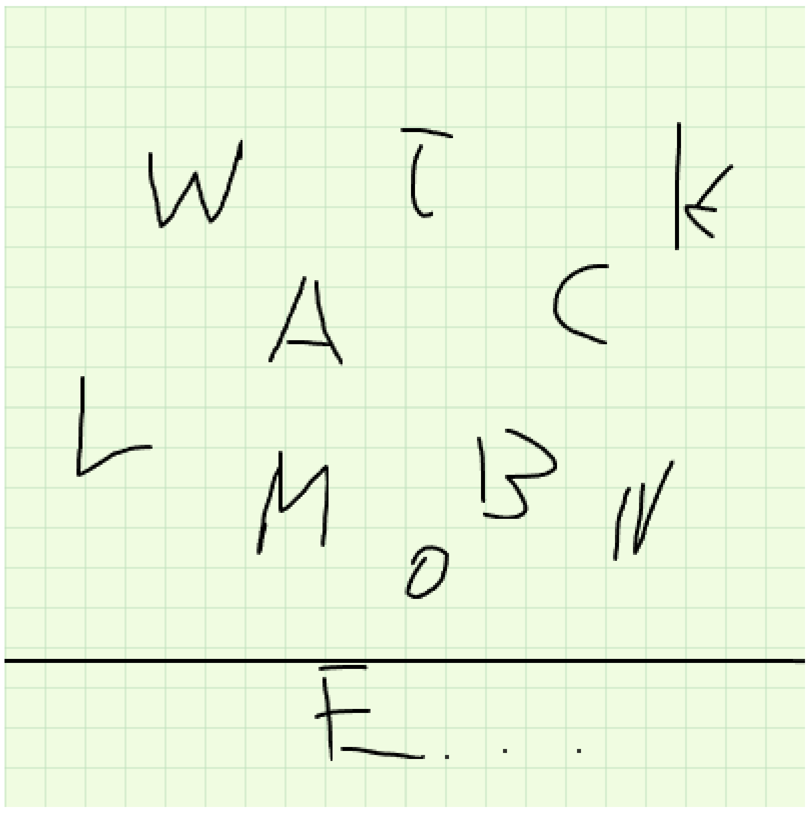
A word game where letters fall from the top of the screen, and the player must type the correct word that matches the letters to prevent them from reaching the bottom.

Interactions:

- **Typing:** Type the correct word or letter as it falls down the screen.
- **Objective:** Type the word that matches the falling letters before they hit the bottom.
- **Rhythm game**: It will be alot like a rhythm game just without the music

Main Design Elements:

- **Start Screen:** There will be a start screen for the game
- **Letters:** Random letters fall from the top.
- **Colors:** The colors will be
- **Words:** As letters fall, they form words that the player must type.
- **Score:** Points are awarded for typing words correctly.
- **Game Over:** The game ends when the letters pile up to the bottom of the screen.



Idea 5: Bounce the Ball

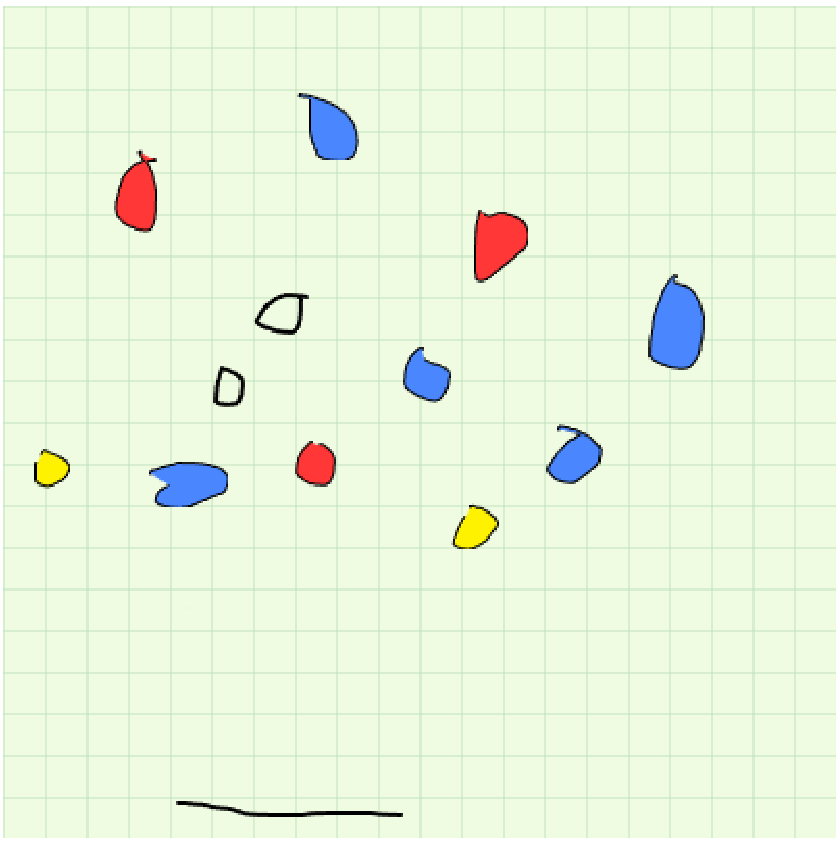
Main Idea:

A simple physics-based game where you control a paddle to bounce multiple balls with different physics applied to them. Each ball will have a unique ability and they will despawn over time

Interactions:

- **Movement:** Use the mouse to move the paddle horizontally.
- **Objective:** Keep the ball in play and break all the blocks.
- **Collision:** The Ball will collide with paddle

Main Design Elements:

- **Balls:** The balls bounce off walls and the paddle, each ball will have different abilities on it.
- **Paddle:** A horizontal paddle that you control with the mouse to bounce the ball.
- **Score:** Points for each time a ball bounces off of the paddle
- **Lose:** When 10 or more balls reach the bottom of the screen
- 
.

Idea 6: Gravity Ball

Main Idea:

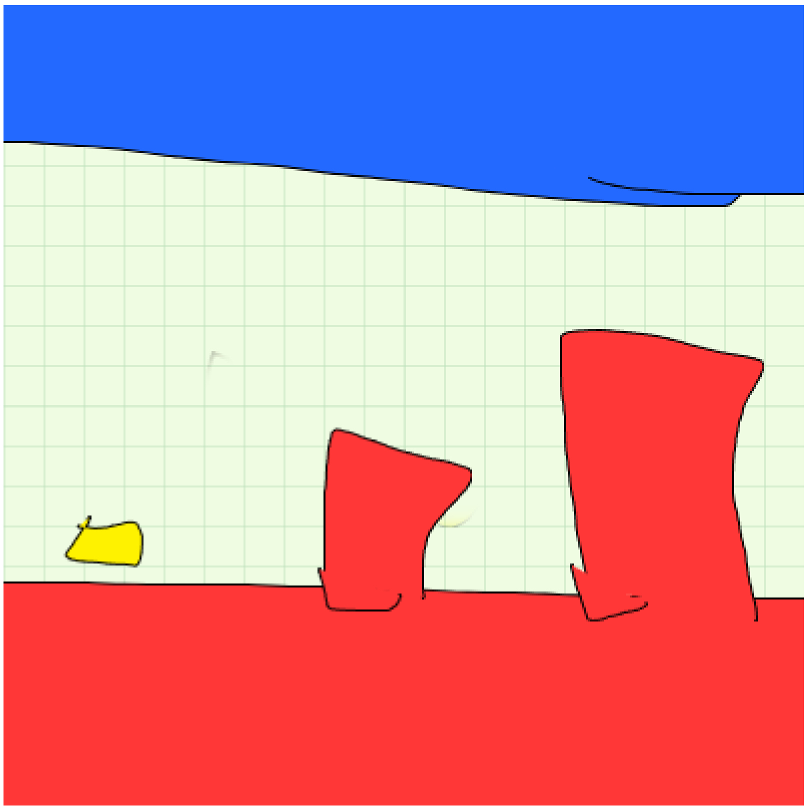
A physics-based puzzle game where you control the gravity of a ball to navigate through levels and reach the goal.

Interactions:

- **Movement:** Use arrow keys to move the ball in normal gravity.
- **Gravity Control:** Press the spacebar to reverse gravity, making the ball move in the opposite direction.
- **Objective:** Reach the goal by manipulating gravity to avoid obstacles.

Main Design Elements:

- **Ball:** A simple ball that moves left or right based on gravity.
- **Gravity:** The gravity direction can be switched with the spacebar, making the ball move up or down.
- **Obstacles:** Walls, spikes, and moving platforms that challenge the player's timing.
- **Goal:** A target or goal point the player must reach.



Bounce the Ball Game

1. Game Initialization

- **Set Up the Game Area**:
    - Set the screen dimensions (e.g., width, height).
    - Place the paddle at the bottom of the screen.
    - Create a list of balls with random properties (speed, angle, color, and unique abilities).
    - Initialize variables for tracking score and the number of balls lost.

2. Main Game Loop

- **Game Loop**: This loop will run until the game is over. In each iteration:
    - Update the paddle's position based on mouse input.
    - Update the ball positions and check for collisions with walls, the paddle
    - Track the score by counting how many times a ball bounces off the paddle.
    - Track the number of balls that fall below the screen, marking the player as "Game Over" when 10 or more balls are lost.

3. Paddle Movement

- **Follow Mouse**: The paddle will follow the horizontal position of the mouse. The vertical position remains fixed at the bottom of the screen.
- **Constrain Paddle**: Ensure that the paddle does not go off the screen edges.

4. Ball Movement and Physics

- **Ball Movement**: Each ball moves in a straight line, influenced by speed and angle.
    - Balls will bounce off the walls (left, right, top) and the paddle.
    - The direction of movement will reverse when the ball hits the walls or the paddle.
    - Balls will have unique behaviors (random speed, special abilities like split, increased speed after bounce, etc.).

5. Ball Collision Detection

- **Block Collision**: When a ball hits a block, the block is destroyed, and the ball changes direction.
    - Blocks should disappear when hit by a ball, and the player earns points.
- **Paddle Collision**: When a ball intersects with the paddle, it will bounce off and speed up slightly.
    - Increment the score each time a ball bounces off the paddle.

6. Game Over Condition

- **Track Lost Balls**: Every time a ball goes below the screen, it is considered lost.
    - The game ends when 10 or more balls are lost.

8. Restart or Quit

- **Option to Restart or Quit**: After the game ends, give the player an option to restart the game or quit.

Summary of Pseudo Code Flow

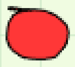
```
initialize game:
set up screen, paddle, blocks, balls, and score
while game is running:
    -   update paddle position based on mouse
    -   for each ball:
    -    move ball, check for collisions with walls, paddle, blocks
    -   if ball hits paddle, bounce and increase score
    -   if ball hits block, destroy block, bounce, and increase score
    -   if ball falls off screen, increment lost_balls
    -   if lost_balls >= 10, end game
    -   render screen (paddle, balls, blocks, score)
if game over:
    -   display score and ask to restart or quit
```

Additional Game Features

- **Power-Ups**: Introduce power-ups that appear occasionally after a block is destroyed (e.g., enlarge paddle, slow down balls).
- **Ball Abilities**:
    - Some balls may have special abilities (e.g., faster speed after bouncing, splitting into smaller balls upon hitting the paddle).

1/10 [illegible] 2°°

Fost
Blits

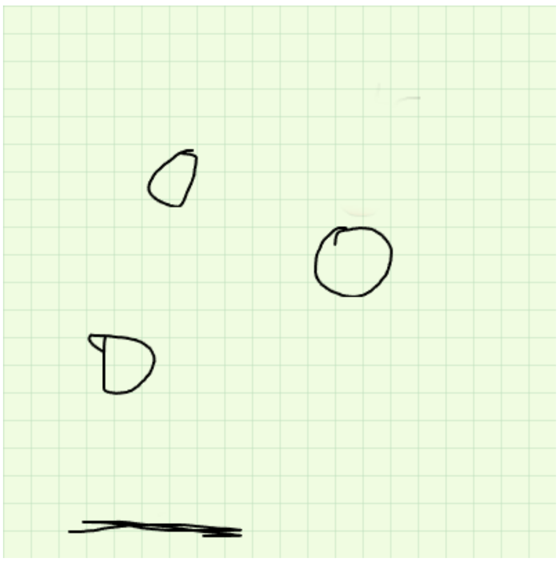


Galde

Implementation Log

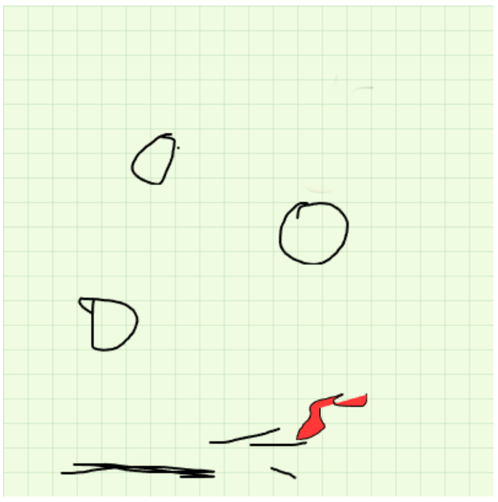
Mistake #1: Frame Rate Freezing Inconsistently

Issue:
When I implemented the "Stop Time" ability by setting frameRate(0), then I realised that it was stupid since I really only wanted the balls to stop. Also freezing the game fully is super finicky so I decided to not use that. Originally I had a similar clock system to the ball timer but it for some reason isn't working.



Nothing Here is moving and everything is frozen, everything is also gonna be frozen forever

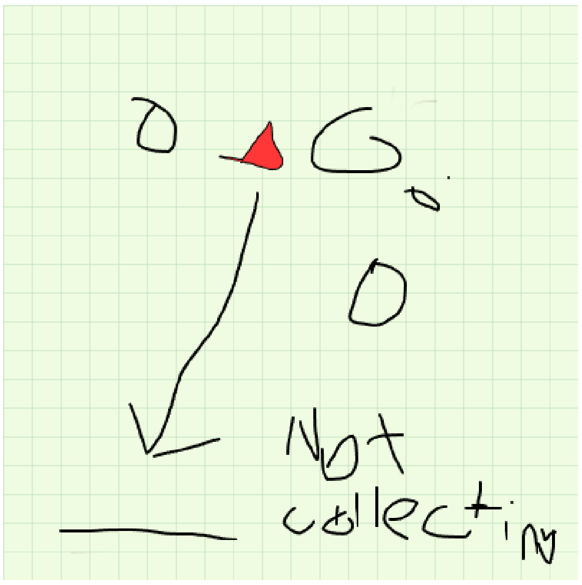
Solution:
To solve this I was like screw it, i'll just use the same code for the ball slowing ability but ill make the ball super super slow. Unsurprisingly this worked exactly as I intended it to work.



Now as we can see the paddle can still move but the balls can not

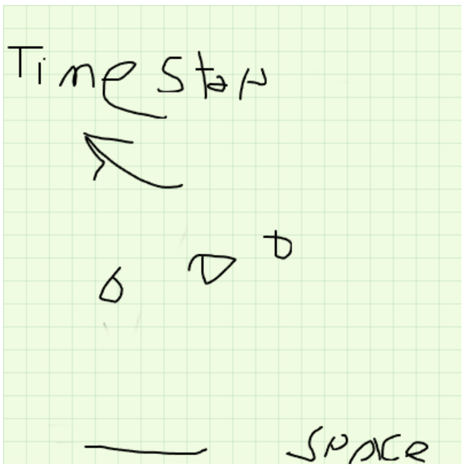
Mistake #2: Power-Up Not Being Collected

Issue:
The power-up was not being collected when it touched the paddle or when the player interacted with it. It wasn't triggering the collected flag properly, and sometimes the power-up would stay on screen indefinitely or not activate when collected.



As we can see it's going to go to the bottom thing but it simply just doesn't want to collect for some reason

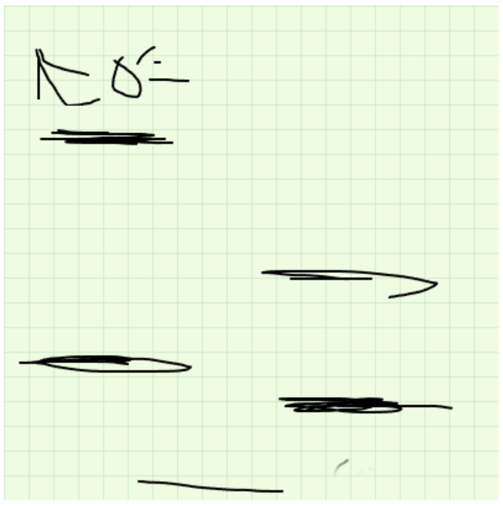
Solution:
Soooo instead I just completely removed the feature and decided to just make it so that you get a random power up every 5-10 seconds of gameplay. And they can be used as long as their availability is still there.



Now we can see that unlike before the paddle can move and the balls move such a small amount its not noticeable

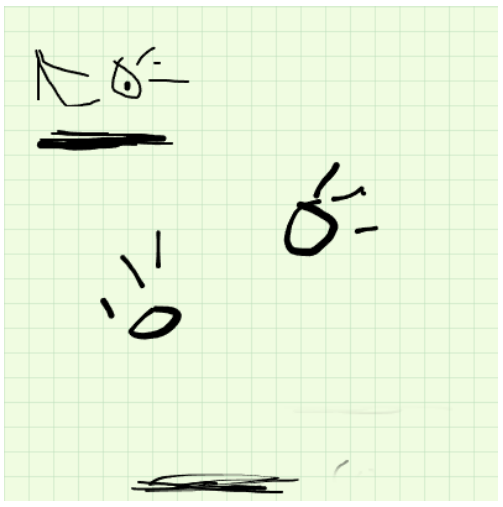
Mistake #3: Wall Despawning Logic Not Working

Issue:
The wall spawned using the power-up did not despawn after the allotted time. The shouldDespawn() function was not properly checking whether the wall had expired, causing the walls to remain indefinitely in the game, cluttering the screen.



As we can see there are multiple walls because of the fact that they are not despawning properly

Solution:
I checked the logic in the shouldDespawn() function and realized that the time tracking was not properly updating. I fixed it by ensuring that I updated the wall's lifespan correctly and checked against the current time to determine whether it should be removed. I then added code to remove the wall from the walls list once its lifespan expired. Basically making it exactly the same as the ball despawn logic.



Now we can see that there is only one wall left because all the other walls have despawned properly.